Adaptive Classification Under Computational Budget Constraints Using Sequential Data Gathering

Joachim van der Herten[a,*], Ivo Couckuyt[a,1], Dirk Deschrijver[a], Tom Dhaene[a]

[a] *Ghent University - iMinds, Technologiepark 15, B-9052 Gent, Belgium*

Abstract

Classification algorithms often handle large amounts of labeled data. When a label is the result of a very expensive computer experiment (in terms of computational time), sequential selection of samples can be used to limit the overall cost of acquiring the labeled data. This paper outlines the concept of sequential design for classification, and the extension of an existing state-of-the-art research platform for surrogate modeling to handle classification problems with sequential design. The capabilities of the platform are illustrated on a number of use cases including real-world applications such as an ElectroMagnetic Compatibility (EMC) and a Computational Fluid Dynamics (CFD) problem. The CFD problem also illustrates how classification can be used together with regression techniques to solve multi-objective constrained optimization problems of complex systems.

Keywords:
Classification, SUMO Toolbox, Sequential design, Surrogate modeling

1. Introduction

Supervised learning algorithms learn the relation between an input space and a corresponding output space based on multiple examples (*samples*).

*Corresponding author

Email addresses: joachim.vanderherten@ugent.be (Joachim van der Herten), ivo.couckuyt@ugent.be (Ivo Couckuyt), dirk.deschrijver@ugent.be (Dirk Deschrijver), tom.dhaene@ugent.be (Tom Dhaene)

URL: http://sumo.intec.ugent.be (Tom Dhaene)

[1] Ivo Couckuyt is a post-doctoral research fellow of the Research Foundation Flanders (FWO)

After learning, the predictor can be used to predict the output(s) of unseen data points. In case an output varies continuously, this task is referred to as *regression*. When only a distinct number of discrete outcomes are possible (*labels*), the term *classification* is used. In literature, classification algorithms usually label large data sets. To limit the massive computational requirements of the learning process, the data is often sub-sampled to obtain a smaller representative set of training data.

Sometimes, obtaining the label for a sample is a very *expensive* task: it might be the result of a lengthy computer simulation or a (possibly dangerous) real-life experiment. Assuming there are *budget constraints* limiting the total amount of labels that can be acquired, obtaining the labels for all samples in the data set might not be possible. Although budget constraints also include applications where time and money is required for instance preparation [1], this article focuses on labels obtained through evaluation of complex physics-based (deterministic) simulators. These are used frequently in computer-aided design and engineering (CAD / CAE) to avoid building and testing several prototypes of new products. As these simulations have become significantly more accurate over the years, their computational requirements have also become more expensive.

This article describes a state-of-the-art platform for surrogate modeling [2] with sequential design. A surrogate model is a cheap-to-evaluate mathematical regression model mimicking the response of computationally intensive simulators with continuous response range, and are trained from a small set of (sequentially) well-chosen evaluations. The platform was recently expanded with classification models and some state-of-the-art sequential design methods targeting classification applications. The SUMO Toolbox is introduced in Section 2 with a focus on these new extensions. The concept of sequential design is introduced in Section 3, and the sequential sampling step for classification is discussed in more detail in Section 4. The integrated platform is then illustrated on a number of use cases in Section 5.

2. SUMO Toolbox

Designed as a research platform for sequential sampling and adaptive modeling using MATLAB, the SUMO toolbox [3] has grown into a mature design tool for surrogate modeling with sequential design offering a large variety of algorithms for simulators with continuous output. The software design is fully object-oriented allowing high-extensibility of its capabilities.

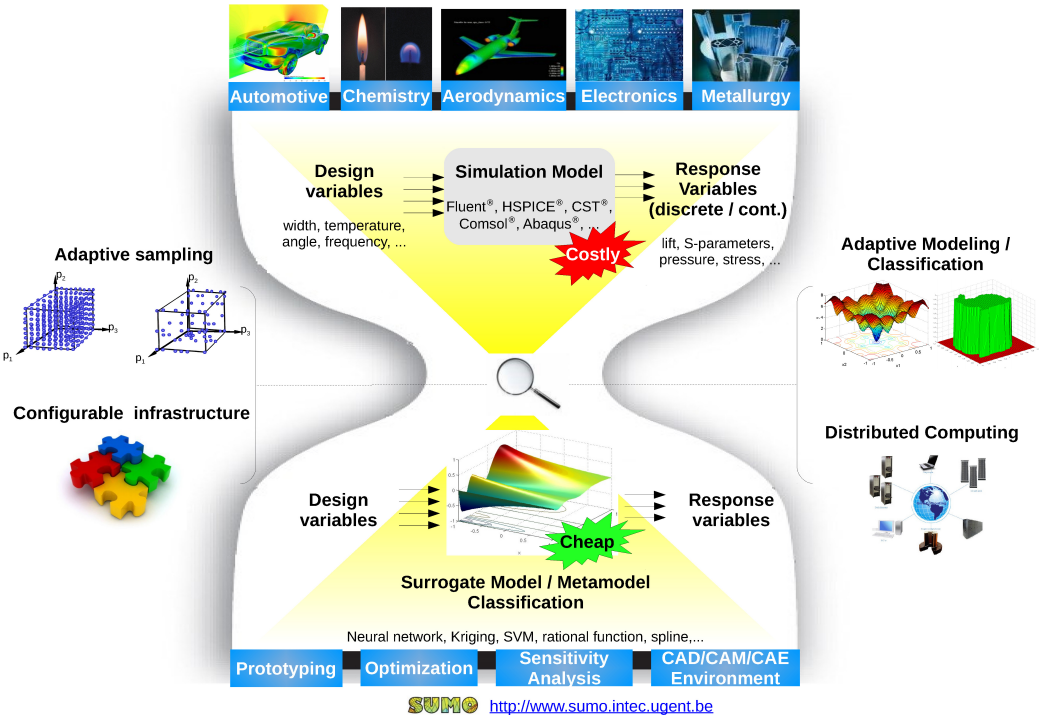


Figure 1: Design philosophy of the SUMO Toolbox for surrogate modeling. The toolbox was recently extended to support classification applications under budget constraints.

By default, the platform follows the integrated modeling flow as shown in Figure 3, but can also be configured to model data sets, use a one-shot setup etc. Recently, the platform has been extended to offer support for several classification algorithms by including several implementations and linking the WEKA library [4].

Figure 1 illustrates the design goals of the SUMO Toolbox. Expensive computer simulations of complex black-box systems with several design parameters are approximated by a cheap-to-evaluate model, and the toolbox can also approximate outputs with a discrete set of labels by training a classifier. To obtain these goals, the SUMO Toolbox offers sequential sampling and adaptive modeling in a highly configurable environment which is easy to extend due to the microkernel design philosophy as illustrated in Figure 2. Distributed computing support for evaluations of data points is also available, as well as multi-threading to support the usage of multi-core architectures for regression modeling and classification.

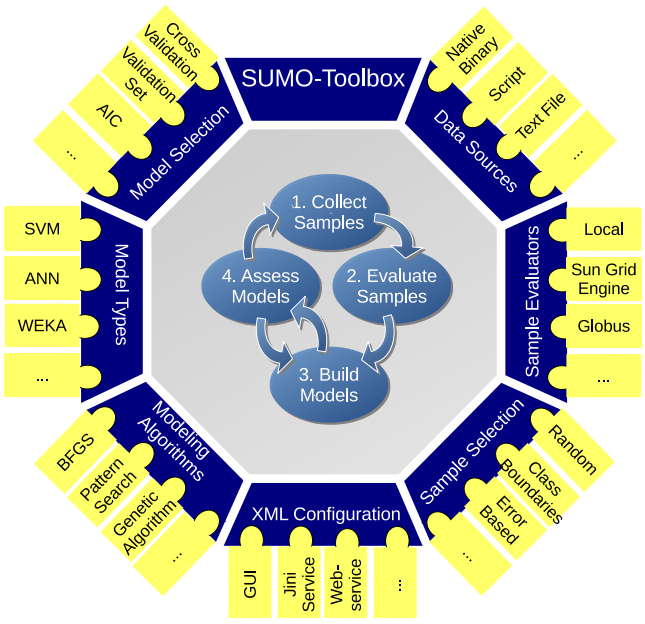


Figure 2: Microkernel architecture of the SUMO Toolbox.

Many different plugins are available for each of the different sub-problems: model types (rational functions, Kriging [5], splines, Support Vector Machines (SVM) [6, 7, 8], Artificial Neural Networks (ANN), Extreme Learning Machines (ELM) [9], Least Squares-SVM (LS-SVM) [10], Random Forests [11]), hyperparameter optimization algorithms (Particle Swarm Optimization [12], Efficient Global Optimization [13], simulated annealing, Genetic Algorithm), sample selection (random, error based, density based [14, 15], hybrid [16]), Design of Experiments (Latin hypercube [17, 18], Box-Bhenken), and sample evaluation methods (local, on a cluster or grid). The behavior of each software component is configurable through a central XML file and components can easily be added, removed or replaced by custom implementation.

During the adaptive modeling step, the Toolbox uses the following methodology for model selection to guide the hyperparameter optimization: the quality of a model $\tilde{f}_\theta$ parametrized by θ of a dataset D is denoted as:

$$\Lambda\left(\epsilon, \tilde{f}_\theta, D\right). \tag{1}$$

Λ denotes a quality estimator for model selection: the SUMO Toolbox supports several algorithms such as a validation set, cross validation, Akaike Information

Criterion (AIC) [19], SampleError, jack-knife and LRM [2]. The quality estimator uses an error function ϵ: popular choices are Root Mean Square Error (RMSE), Root Relative Square Error (RRSE) for regression [20], or the misclassification rate for classification.

The architecture for hyperparameter optimization of the toolbox also allows optimization of the classifier parameters to improve its position in the ROC space, a popular method to present the accuracy of a classifier. This can be seen as a multi-objective goal: minimizing the false positive rate and maximizing the true positive rate. Both rates can be determined by evaluation of a quality estimator. By combining these objectives into a single multi-objective measure Λ, the hyperparameter step becomes a multi-objective optimization problem which is supported. This results in a set of pareto-optimal solutions representing the trade-off between both objectives, instead of a single optimal solution [21].

The SUMO Toolbox is free for academic use and is available for download at http://sumo.intec.ugent.be. It can be installed on any platform supported by MATLAB. In addition, a link can be found to the available documentation and tutorials to install and configure the toolbox including some of its more advanced features. News items concerning new releases, additional features and updates can also be found at the same web page.

3. Sequential Design

This section describes the concept of sequential design, the default experimental design method of the toolbox, and explains how it differs from one-shot experimental design [14, 15, 2] methodology as used traditionally in the design and analysis of computer experiments. This methodology can be applied when the output of the computer experiments is continuous (regression, as usually encountered in surrogate modeling) and for discrete outputs (classification) as described in this article. The usage of *model* in this section covers both the regression and classification interpretation, unless specified otherwise. This section first reviews one-shot and sequential experimental design, and then lists some categories of existing methods for sequential design.

3.1. One-shot experimental design

The selection of data points is of key importance when obtaining the output is expensive. Each evaluation should reduce the model uncertainty as much as possible, and should contribute a maximum amount of information.

This information can be obtained with a one-shot approach in which the data points are defined by generating an experimental design based on a space-filling criterion[2] at once. Popular methods to generate these designs are (maximin) Latin Hypercubes [17] and factorial designs [23]. All data points are evaluated and a model is trained and evaluated. Because no prior information is available on the behaviour of the response surface it is hard to determine the size of a one-shot design, which is their major downside. In case too few data points have been evaluated to obtain an accurate model (*undersampling*), the process has to be restarted. But the problem can also be easier than anticipated: evaluating more data points than required (*oversampling*) means wasting computational resources.

3.2. Sequential experimental design

Sequential design turns this one-shot approach into an iterative process [24, 3]. The acquired data and the constructed models from previous iterations are analysed in order to intelligently select locations for new data points (sequential sampling). Next, the labels for these additional data points are obtained and new models can be trained or existing models can be updated (in case online learning methods are used to update existing models with additional data [25]). First of all this means there is no risk of over- or undersampling as the process can be halted when the desired accuracy is reached (or if the computational budget is exceeded). A second major advantage is that information provided by the consecutive labels and intermediate models can guide the selection to obtain optimal locations for new data points. This allows the data distribution to be adapted and refined to the problem at hand as more knowledge becomes available, which means the sampling is no longer only guided by space-fillingness. In surrogate modeling, the concept of sequential design has been applied in several succesful applications [26, 27, 28, 24, 29, 30].

Experimental design with sequential sampling is related to the field of active learning [31, 32, 33, 34]. Under its original formulation, active learning picks some data points from a set of unlabeled candidate points for evaluation, after which one or multiple classifiers are trained on the labeled instances. After a performance evaluation, the process may be repeated to label more

[2]Other aspects of Design of Experiments (DoE) such as blocking, replication etc. lose their relevance within the context of computer experiments [22].

training instances to improve the accuracy. Often, an active learning algorithm provides a ranking of possible data points [35]. However, over the years active learning has grown into a broad field which now includes intelligent selection of data points in a continuous space (which implies no predefined set of unlabeled data points is available). Examples include an approach presented by [36] which sequentially learns a decision function, and optimal learning for infinitely many armed bandits problems [37], in which the learner can either sample an arm (or a distribution) that has been already been observed in the past, or sample a new arm with a known mean. There is a reward associated with each arm (each potential sample), and the objective is to maximize the reward and minimize the regret (expected difference between collected reward and the reward associated with the optimal arm). For an excellent overview and mathematical treatment of sequential sampling techniques in active learning, we refer to [38]. The setting considered for sequential sampling in this paper involves sampling near the boundaries of the different classes and balancing exploration and exploitation. The notion of reward or regret is not considered in this work.

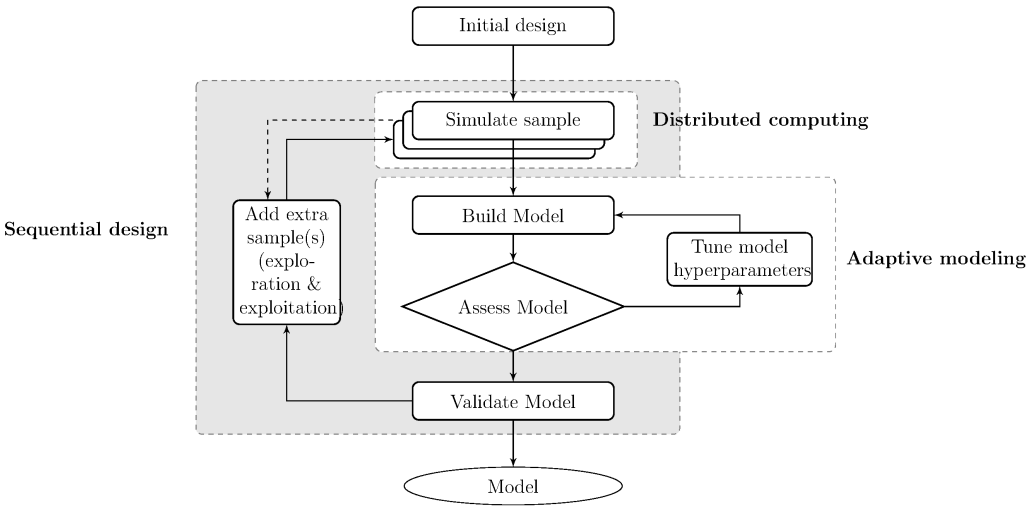


Figure 3: A typical adaptive modeling flow with sequential design. After evaluating an initial set of samples an (intermediate) model is created and tuned. A feedback loop allows for the evaluation of additional samples to improve the quality of the model. When the sample selection strategy is not model-based, sample selection may also proceed without training the intermediate models each iteration as represented by the dashed arrow.

The typical modeling process with sequential design is illustrated in Figure 3: it is initiated by generating a small set of initial data points (referred to as *initial design*) which are simulated. The process then initiates a loop: a model is trained and its *hyperparameters* are optimized with respect to a pre-set quality criterion (discussed in Section 2). When improvement can no longer be realized and the quality of the model is not sufficient, the sequential sampling routine is started. Based on all available information one or possibly more new data points are chosen by this co-routine, of which the labels are acquired. When the labels are available a new model is trained and optimized. This process continues until the stopping criterion is satisfied: either the regression model or classifier is sufficiently accurate, or the budget constraints (maximum number of evaluations or a time limit) are reached.

The sampling and modeling steps of the process are independent (with the exception of modeling-based sampling strategies (Section 4), which means construction of intermediate models is not an absolute requirement (it is possible to immediately select new samples after evaluation as represented by the dashed line in Figure 3): possible scenarios include sequential selection and simulation of samples and construction of a model only when the computational budget is consumed, or selection of samples in batches as opposed to one-by-one.

Continuous outputs can be approximated with regression techniques such as Kriging [5], Artificial Neural Networks (ANN), Radial Basis Functions (RBF) etc. Sequential sampling algorithms typically discover "difficult" regions in the design space and sample them densely as these regions tend to result in high model uncertainty. For regression applications sequential design usually focuses on regions of the design space that are undersampled and where additional samples are needed to discover the response behaviour (input-based exploration), or highly non-linear regions requiring additional information in order to be modeled accurately. The latter requires knowledge on the responses of earlier samples (output-based exploitation).

When discrete outputs are encountered, techniques such as Random Forests (RF) [11], Support Vector Machines (SVM) [6, 7, 8] or Naive Bayes [39] are used to classify the data points in the adaptive modeling step. For these problems, the model uncertainty is usually situated in regions which have been undersampled (exploration), or near the classification boundaries (exploitation).

3.3. Existing sequential sampling methods

An overview of some popular sequential sampling methods for modeling of both regression and classification outputs is given in Table 1. The input-based sequential sampling methods usually focus on expanding one-shot designs (such as the nested Latin Hypercubes) or (Quasi-) Monte Carlo methods. It is also possible to optimize space-filling criteria (for instance, maximin or minimax) as in [15].

The output-based sequential sampling methods analyze the obtained labels or output values in order to guide the selection of new data points. They can pursue optima or focus on non-linear areas. These methods are typically complemented with an input-based method to ensure not too much focus is put on the *exploitation* of the acquired knowledge. All listed algorithms include an input-based component to ensure exploration as well, and are popular methods for building designs with several successful applications [24, 29, 40, 30, 41].

A third type of sequential sampling methods directly query the intermediate models built during the adaptive modeling phase. These methods can rely on the ability of some model types to explicitly indicate regions of high uncertainty (for example the prediction variance of Kriging [5], or the probabilistic SVM [8]). It is also possible to train several models and find the regions with most disagreements (*query by committee* methods [42, 43, 44]). This results in model-based sampling approaches. This type of methods, however, creates a dependency between the sampling and modeling steps (and often comes at a non-negligible extra computational cost). Some model-based sampling strategies are listed in the third column. Model error sampling for instance, samples in areas with most disagreements between the model and the actual outputs [45]. Model-based methods typically pursue a specific goal: EDSD (Explicit Design Space Decomposition) [46] for instance is a sequential design method for refining the class boundary of an SVM model, whereas Probability of Feasibility [47] searches for areas which exceed a certain threshold making them suitable for sampling constrained areas.

4. Sequential sampling for classification

We now review some aspects of sequential design for classification problems approximating a black-box simulator with discrete outputs. Two sequential sampling approaches are described in more detail: Neighbourhood-Voronoi [54]

Table 1: Categorization of Sequential designs.

INPUT-BASED	OUTPUT-BASED	MODEL-BASED
Low discrepancy sequences [48, 49, 50]		Probability of Feasibility (PoF) [47]
Sequentially Nested Latin Hypercubes [51, 52, 17]	(F)LOLA-Voronoi (Regression) [16, 53]	Model error sampling (regression) [45]
Monte-carlo/Optimization Based [14, 15]	Neighbourhood-Voronoi (Classification) [54]	D- and G-optimal designs [55, 56]
Voronoi-based [16]		Explicit Design Space Decomposition (EDSD) [46]
Random		Sequential Exploratory Experimental Design (SEED) [42]

(an output-based approach) and Probability of Feasibility [47] (a model-based approach). These methods are also applied to the test cases in Section 5.

4.1. Neighbourhood-Voronoi

By default, the SUMO Toolbox offers the earlier introduced Neighbourhood-Voronoi (N-V) algorithm [54] for classification, a sequential sampling strategy combining exploration and exploitation for the construction of accurate classifiers. This algorithm is a modification of the LOLA-Voronoi [16] sequential sampling algorithm used in surrogate modeling. The Neighbourhood-Voronoi algorithm is based on the Voronoi tesselation of the search space and focuses on two distinct goals:

- Discover the class regions: the input space should be explored to find the (sub-)regions of the different classes. When nothing is known about the problem at hand, the choice of new data points should be influenced by the possible existence of undiscovered regions. As iterations evolve and all (possibly disconnected) regions of all classes have at least one

data sample the *exploration* can be halted. Depending on the problem, this knowledge may be available or not.

- Refine the boundaries: when two or more distinct regions have been identified, new data points should be chosen such that the location of the boundary between the regions can be identified. This *exploitation* component greatly enhances the accuracy of the classifier.

For each data point $\boldsymbol{p}_r$ of a set of data samples P, the N-V algorithms first selects a set of nearby points $N(\boldsymbol{p}_r) \subset P \backslash \boldsymbol{p}_r$ known as the *neighbourhood*. The choice of neighbouring points is guided by two principles: the neighbourhood should have a high cohesion (defined as the average distance of the points in $N(\boldsymbol{p}_r)$ and $\boldsymbol{p}_r$) and low adhesion (the average minimum distance of points in $N(\boldsymbol{p}_r)$ from each other). Clearly, these two principles conflict as a higher cohesion implies higher adhesion as well. When the size of the neighbourhood equals twice the dimensionality of the data samples, the optimal configuration is known as the *cross-polytope*. A candidate neighbourhood is first assigned a score which indicates how much it resembles to a cross-polytope:

$$R(N(\boldsymbol{p}_r)) = \frac{A(N(\boldsymbol{p}_r))}{\sqrt{2}C(N(\boldsymbol{p}_r))}. \tag{2}$$

To obtain the neighbourhood score which is used to guide the search amongst all possible neighbourhood candidates, R is divided by C to prefer neighbourhoods with low cohesion if two candidates are found which resemble the cross-polytope configuration equally:

$$S(N(\boldsymbol{p}_r)) = \frac{R(N(\boldsymbol{p}_r))}{C(N(\boldsymbol{p}_r))}. \tag{3}$$

Once the neighbourhood candidate with optimal S has been selected, the labels of the points in $N(\boldsymbol{p}_r)$ are compared: when no disagreement is found the Voronoi cell defined by $\boldsymbol{p}_r$ is considered to contain no class boundary. The size of all Voronoi cells is computed and serves as a basic score. If a disagreement is found in the labels of the points in $N(\boldsymbol{p}_r)$, the score is increased. New samples are then selected within the Voronoi cells with highest score: this could be because a disagreement was found and the cell is large compared to other cells with disagreements (*exploitation*) or because the cell became very large compared to all other cells and should be sampled, even if

a disagreement has not yet been found (*exploration*). A full description of the N-V algorithm can be found in [54].

The N-V algorithm is an excellent choice for sequential selection of data points. The method can define all data points upfront, independently of classifiers to be trained in a later step: N-V does not query the classifier for regions of uncertainty. The benefit of having a sampling strategy independent from the intermediate classifiers is significant when only a small number of data points have already been evaluated: at this point the classifier is still unstable because it lacks information which might influence the sample selection undesirably. Furthermore the N-V algorithm automatically balances exploration and exploitation which allows discovery of previously undiscovered class regions. The latter property distinguishes N-V from other methods such as EDSD [46], which assumes the initial set of points finds all regions. A downside of the N-V algorithm is its increasing computational complexity as the dimensionality of the input space grows (similar to the problems encountered with the LOLA-Voronoi algorithm). However, this issue could be tackled by applying a faster method to select $N(\boldsymbol{p}_r)$. Current research investigates the use of the strategy proposed in [53] for the N-V algorithm.

4.2. Probability of Feasibility

A model-based method for sequential design is the Probability of Feasibility (PoF) [47]. This criterion picks new data points in underexplored areas which have a high probability of remaining below a certain threshold $g_{\min}$. Formally this denoted as

$$P(F(\boldsymbol{x}) < g_{\min}) = \Psi\left(\frac{g_{\min} - \tilde{f}(\boldsymbol{x})}{\tilde{s}(\boldsymbol{x})}\right). \tag{4}$$

The PoF is typically used with Kriging or Gaussian process models, represented by random variable $F(x)$, with prediction mean $\tilde{f}$ and variance $\tilde{s}$. The function Ψ corresponds to the cumulative density function of the standard normal distribution. For classification problems, the PoF can be interpreted as the probability estimate of a probabilistic classifier. be used in combination with probabilistic classification models such as the probabilistic SVM. This approach is very suitable for modeling constraints when the output of the constraints is discrete (feasible/infeasible).

5. Test cases

5.1. Stanford Bunny

In this illustration, a classifier is trained for the Stanford Bunny 3D model [57] consisting of 69451 polygons. The input space is three-dimensional (x, y, z coordinates) and the output is binary: a zero indicates the point is outside of the model, a one indicates the point is inside. The resulting class boundary is the contour of the object. In fact, checking if a point is inside or outside of an object is not a very computationally complex task: in this article it is only used to illustrate the capabilities of the sequential approach, as well as the toolbox.

The toolbox was configured with an initial Latin Hypercube generated by the Translational Propagation algorithm [18] (50 points). Each iteration of the sequential design, 10 additional points were selected by the Neighbourhood-Voronoi algorithm. The process was terminated when 1000 samples were evaluated. Given the shape of the 3D object, this is quite a sparse data set (the size corresponds to a $10 \times 10 \times 10$ grid). The growth and evolution of the dataset[3] is illustrated in Figure 4.

For each iteration, several classifier types (SVM, ANN, Random Forests and Naive Bayes) were trained concurrently in several threads to evaluate the performance of each classifier for this application. For the SVM, the DIRECT algorithm [58] was used to optimize the kernel parameter (RBF kernel) and the regularization parameter. For ANN, a Genetic Algorithm (10 generations of 15 individuals) was used to optimize the network architecture and initial weights. Each individual network was trained with Levenberg-Marquard backpropagation with Bayesian regularization (300 epochs) [59]. Random Forest (fixed number of 500 trees) and Naive Bayes had no parameters to be optimized.

For hyperparameter optimization, 5-fold crossvalidation was used as performance measure. In addition, the classifiers were also validated on a dense validation set to estimate their true error. A common problem specific to quality estimation of classifiers is caused by *class imbalance*. If a class is underrepresented, a straightforward error function such as the miss-classification rate will favor classifiers discriminating the minority class, because the labels of the majority class are mostly predicted correctly. In extreme cases, the

[3] A short movie of the sequential selection of data points can be seen on https://www.youtube.com/watch?v=EcvfbaSUMOw

minority classes will be completely ignored. To avoid this, the geometric average of the *precision* and *recall* of both classes (out, in), represented by $p_{\text{out}}, p_{\text{in}}$ and $r_{\text{out}}, r_{\text{in}}$ respectively, is used as error function for this experiment:

$$G = \sqrt[4]{p_{\text{out}} p_{\text{in}} r_{\text{out}} r_{\text{in}}}. \tag{5}$$

Let A denote the set of data points with label A, and $\Pr(A)$ the set of data points labeled A by the classifier. We define the set of *true positives* as $\text{TPF}(A) = A \cap \Pr(A)$. The recall of the classifier for the class A is the ratio of true positives and the number of data points with label A:

$$p_A = \frac{|\text{TPF}(A)|}{|A|}, \tag{6}$$

whereas the precision of A is defined as the ratio of the true positives and the number of predicted cases of A:

$$r_A = \frac{|\text{TPF}(A)|}{|\Pr(A)|}. \tag{7}$$

A score of $G = 1$ represents a perfect classifier as it implies all precision and recall terms have a value of 1. This means the classifier labels all data points correctly, a score of $G = 0$ represents a missclassification of every data point [60].

Figure 5 shows the obtained G-score on the validation set as more samples are evaluated, and classifiers are retrained. The classifier accuracy improves as the number of samples increases for all methods included in this illustration: RF, SVM and ANN are performing very similar, but SVM always seem to be slightly better. The results for the ANN show most fluctuation: closer inspection reveals the optimization of the network architecture sometimes gets stuck in a solution which scores well for crossvalidation, but performs worse on the validation set: when new data points are added, the crossvalidation score drops and the network architecture needs to be altered. This causes the bumpy behaviour of the ANN performance. Naive Bayes clearly is not suited to model the boundary of the 3D model: its score G score is stuck around 0.6 and is barely increasing as additional data points are added. Of all methods it performs worst.

Considering $G = 0.9$ corresponds to a very satisfying classifier for this application, SVM obtains the score after 200 evaluated samples. In comparison, an SVM trained on a one-shot maximin Latin Hypercube of 200 points

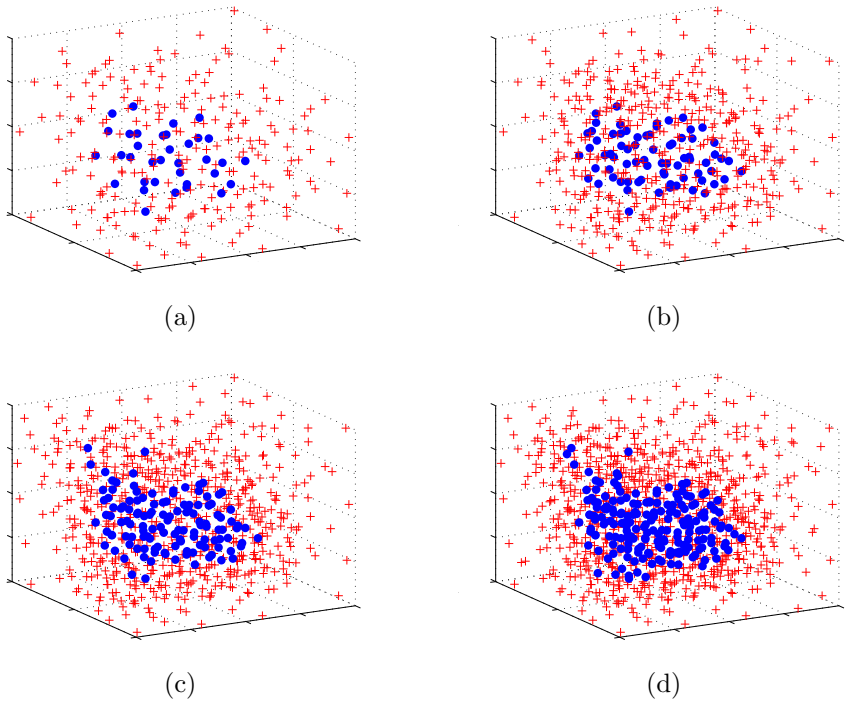


Figure 4: Stanford bunny: Evolution of the samples used to train the classifiers at 250, 500, 750 and 1000 samples in Figures (a), (b), (c) and (d) respectively. Blue dots are inside the 3D object, red crosses are outside.

generated by the Translational Propagation algorithm [18] obtains a score of only $G = 0.85$. The final best SVM Model was evaluated on a dense grid and the obtained labels were used to generate an iso-surface of the Stanford Bunny which is shown in Figure 6.

5.2. Bended microstrip

This section describes the use of adaptive classification in the field of electromagnetic compatibility (EMC) [54]: the *near-field* (NF) pattern of a double bended microstrip line that was measured using a scanning system as illustrated in Figure 7. The printed circuit board (PCB) comprises a microstrip on a substrate. The microstrip was excited with a generator set and the amplitude of a field component, e.g. $|H_y|$, was measured with an NF scanner of which the head can be moved automatically in two dimensions at a fixed height of 2mm above the Device Under Test (DUT) to perform the

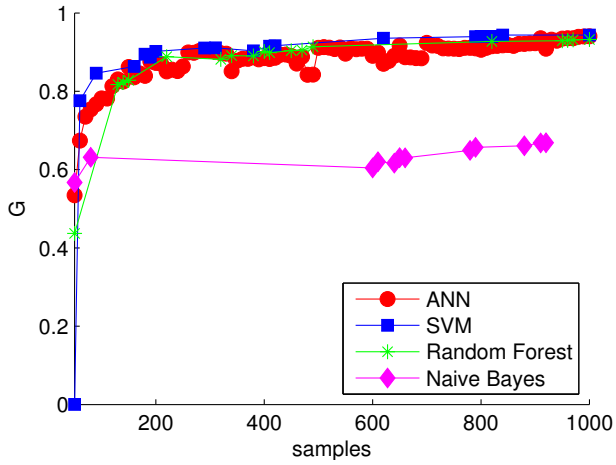
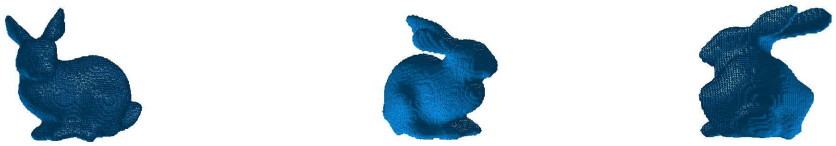


Figure 5: Stanford Bunny: The evolution of the geometric mean of the precision and recall of both classes on the validation set for all classifier types as more samples are evaluated (up to 1000).



(a) (b) (c)

Figure 6: Stanford bunny: A grid of 10^7 points was classified by the final SVM model of the Bunny based on 1000 data samples. Iso-surface techniques were use to plot a volume using the resulting labels. Clearly, the SVM manages to fit the contour of the model very accurately.

measurements.

The NF pattern is a continuous output, however we would like to identify radiation hotspots, regions with elevated radiation, and areas with low radiation near the board. Table 2 indicates how the output range was mapped onto these three labels. A small latin hypercube design of 30 points generated by the Translational Propagation algorithm [18] was used as initial design. The input space consists of the the (x,y)-coordinate on the PCB. Each iteration the Neighbourhood-Voronoi algorithm selects a new sample. After evalua-

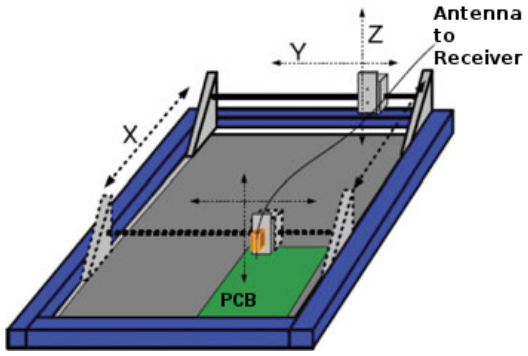


Figure 7: Bended microstrip: near-field scanner setup.

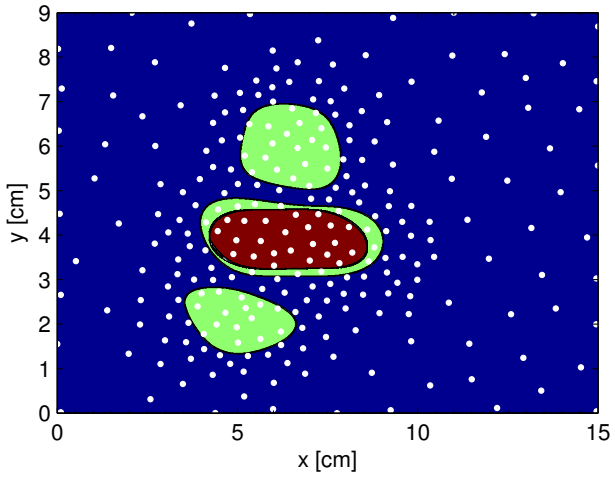


Figure 8: Bended microstrip: contour plot of the final SVM classifier based on 264 measurements (dots). The colors of the classes correspond to the last column in Table 2. The focus of the sampling algorithm is on the class boundaries near the hotspot region.

tion, an SVM classifier (RBF kernel) with two parameters (kernel bandwidth and regularization parameter) optimized by the DIRECT algorithm [58] was trained on the dataset, the performance of the classifiers was estimated by crossvalidation. A simple error function such as the miss-classification rate results in a premature end of the process due to class imbalance. To counter this issue, the geometric average of precision and recall for all three classes was used.

When 264 measurements were evaluated, the desired accuracy of $G = 0.90$

Table 2: Bended microstrip: partitioning of the NF Range in three different classes.

Class Label	NF Range (dB μV)	Color
Low	[0 - 30]	Blue
Elevated	[30 - 35]	Green
High	[35 - inf[	Red

(a score of 1 represents a perfect classifier with perfect precision and recall for all classes) was obtained and the process was halted. Figure 8 shows a plot of the distribution of labels of the final classifier, and all measurements as chosen by the sequential design strategy. A strong focus is on the region containing the hotspot: it is surrounded by a thin region with elevated radiation which requires high sampling density to obtain sufficient information on the class boundaries. This concentration effect did not cause the central region to be oversampled. The exploration part of Neighbourhood-Voronoi has explored the design space to avoid missing out a class region: if any region was missed it is no larger than the size of the largest Voronoi cell.

5.3. Cyclone optimization

The adaptive classification strategy can also be used to model computationally expensive black-box constraints in optimization problems. In this section a 7D constrained Computational Fluid Dynamics (CFD) design problem is studied. Multi-Objective Surrogate Based (Bayesian) Optimization (MOSBO) [61] is used to find pareto-optimal solutions. Gas cyclones are widely used in air pollution control, gas-solid separation for aerosol sampling and industrial applications when large particles are to be caught. In cyclone separators, a strongly swirling turbulent flow is used to separate phases with different densities. A tangential inlet generates a complex swirling motion of the gas stream, which forces particles toward the outer wall where they spiral in the downward direction. Eventually the particles are collected in the dustbin (or flow out through a dipleg) located at the bottom of the conical section of the cyclone body. The cleaned gas leaves through the exit pipe at the top. The cyclone geometry [62] is described by seven geometrical parameters: the inlet height a, width b, the vortex finder diameter D_x, and length S, cylinder height h, cyclone total height H_t and cone-tip diameter B_c. Modifying these parameters has an impact on the gas cyclone itself. Two aspects of the cyclone must be optimized: the pressure loss (represented by

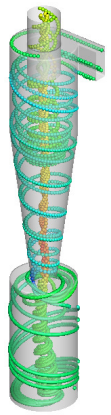


Figure 9: Cyclone: illustration of a cyclone separator.

the Euler number) and the cut-off diameter. The latter is represented by the Stokes number

$$\mathrm{StK} = \frac{t_r\ u}{l}, \tag{8}$$

with t_r representing the particle relaxation time, u the velocity of the fluid away from the obstacle, and l the diameter of the obstacle. The particle relaxation time corresponds to the time constant of the exponential decay of its velocity due to drag.

In addition to both objectives, evaluating the simulator also yields four binary values representing black-box constraints. Each constraint corresponds to internal checks regarding the feasibility of the configuration specified by the user. As each evaluation is computationally demanding this additional knowledge should be included in order to maximize the probability of selecting feasible solutions. Therefore, the constraints should be modeled and included in the optimization process. As the output of the constraints in this example is discrete (feasible/infeasible), we could map both classes to a number (0/1) and apply regression and Probability of Feasibility. However this would essentially be a non-stationary problem (the smoothness of the response surface varies greatly at the boundary) which can lead to problems with Gaussian process and Kriging models [63]. Instead using a probabilistic classification algorithm, we can model the discrete constraint responses and still use the PoF criterion.

To handle this complex 7D multi-objective constrained design problem, the SUMO toolbox is configured to model the Euler and Stokes objectives with Least-Squares SVM (LS-SVM) [10]. The hyperparameters (RBF kernel

bandwidth and regularization parameter) are optimized with the DIRECT algorithm [58]. The sequential design strategy is a combination of two criteria: the Hypervolume Probability of Improvement (HvPoI) [61], a sequential sampling criterion for regression to guide the multi-objective optimization and the Probability of Feasibility (PoF) [47] to guide the optimization towards feasible regions. The combined criterion becomes

$$\gamma(\boldsymbol{x}) = \mathrm{HvPoI}(\boldsymbol{x})\ \mathrm{PoF}(\boldsymbol{x}). \tag{9}$$

The next data point for evaluation is selected by optimizing γ numerically.

To compute the PoF, each constraint is modeled with a probabilistic SVM (RBF Kernel) optimized with the DIRECT algorithm [58]. The quality of the constraint models is assessed by cross-validation, with the F_1-score of the positive class used as error function. The constraints are modeled using the same samples used for training of the surrogate model for the optimization: as the process evolves, the optimization learns the feasibility of the current samples. Inevitably, some samples that violate the constraints will be evaluated while the process evolves. The initial design is a Latin Hypercube of 50 points generated by the Translational Propagation algorithm [18]. Each iteration 5 samples are selected by the sequential design strategy until the sample budget is consumed (120 samples in total).

Figure 10 shows the scores for all evaluated samples for both objectives. The red and green samples form the Pareto front. As the constraints were black-box and were learned throughout the process, many samples have been evaluated that do not satisfy the constraints (as these were not known at that time): only 8% of all 120 samples satisfy the constraints. Fortunately, 4 of them are Pareto optimal and represent valid optimal configurations. The exact optimal Pareto front is unknown, however in order to provide a comparison NSGA-II [64] was applied directly on the CFD simulations for a total of 10000 evaluations: the results are shown in Figure 10. It is clear that the Pareto optimal solutions found by our approach form a similar front to the front found by NSGA-II, however our approach was able to identify these solutions with significantly fewer evaluations. Hence the Pareto front of Figure 10 is a very good approximation given the budget constraint of 120 evaluations.

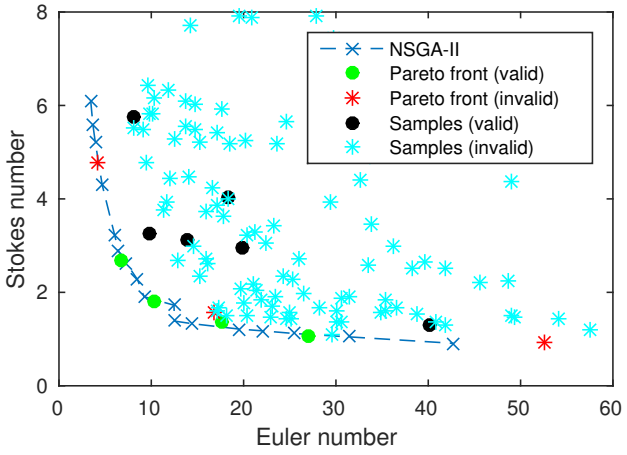


Figure 10: Cyclone: scores for all 120 evaluated samples for the multi-objective cyclone optimization problem. Pareto front points that satisfy the constraints are shown in green, red crosses are Pareto optimal points that do not satisfy the constraints. Black-points are not Pareto optimal, but satisfy the constraints whereas blue crosses are invalid. For comparison, we included the Pareto front obtained by applying NSGA-II on the simulator for 10000 evaluations.

6. Conclusion

The SUMO Toolbox, a state-of-the-art MATLAB Toolbox developed for Surrogate Modeling with Sequential Design has recently been extended to support adaptive training of classifiers, next to its wide variety of regression models. This paper illustrates how the SUMO toolbox can be applied to efficiently solve computational expensive design applications involving classification and optimization problems.

By default, the toolbox uses the sequential design methodology. We discussed the applicability to classification problems with labels resulting from expensive computer experiments. Sequentially, new data samples can be selected to improve the accuracy of the classifier. These new samples are chosen based on what is already known about the application at that point (intermediate classifier, obtained labels, space-fillingness,...).

Improving the sequential sampling algorithms for classification problems (including incorporating existing methodologies from active learning) is subject of further work. In this article, Neighbourhood-Voronoi and Probability of Feasibility are two strategies used for sequential sampling of the class boundaries, both are available in the SUMO Toolbox. We highlighted the

benefits of the Neighbourhood-Voronoi approach, but depending on the classifier and the problem at hand (constraints in optimization, global accurate classifier,...) more optimal strategies can be developed.

Acknowledgment

This research has (partially) been funded by the Inter university Attraction Poles Programme BESTCOM initiated by the Belgian Science Policy Office. The authors thank the Stanford University Computer Graphics Laboratory for the availability of the Bunny 3D model used as illustration in this article.

References

[1] F. J. Ruiz, N. Agell, C. Angulo, SVM-based learning method for improving colour adjustment in automotive basecoat manufacturing, in: ESANN 2009, 17th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 22-24, 2009, Proceedings, 2009.

[2] D. Gorissen, Grid-enabled adaptive surrogate modeling for computer aided engineering, Ph.D. thesis, Ghent University (2010).

[3] D. Gorissen, K. Crombecq, I. Couckuyt, P. Demeester, T. Dhaene, A Surrogate Modeling and Adaptive Sampling Toolbox for Computer Based Design, Journal of Machine Learning Research 11 (2010) 2051–2055, available at http://sumo.intec.ugent.be.

[4] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten, The WEKA Data Mining Software: An Update, SIGKDD Explorations 11 (1). doi:10.1145/1656274.1656278.

[5] I. Couckuyt, T. Dhaene, P. Demeester, ooDACE Toolbox: A Flexible Object-Oriented Kriging Implementation, Journal of Machine Learning Research 15 (2014) 3183–3186.

[6] B. E. Boser, I. M. Guyon, V. N. Vapnik, A training algorithm for optimal margin classifiers, in: Proceedings of the fifth annual workshop on Computational learning theory, ACM, 1992, pp. 144–152. doi:10.1145/130385.130401.

[7] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, ACM Transactions on Intelligent Systems and Technology (TIST) 2 (3) (2011) 27. doi:10.1145/1961189.1961199.

[8] J. C. Platt, Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods, in: J. Smola, P. Barlett, B. Scholköpf, D. Schuurmans, (Editors), Advances in large margin classifiers, Citeseer, 1999.

[9] G.-B. Huang, D. H. Wang, Y. Lan, Extreme learning machines: a survey, International Journal of Machine Learning and Cybernetics 2 (2) (2011) 107–122. doi:10.1007/s13042-011-0019-y.

[10] J. A. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, J. Suykens, T. Van Gestel, Least squares support vector machines, Vol. 4, World Scientific, 2002.

[11] L. Breiman, Random forests, Machine learning 45 (1) (2001) 5–32. doi:10.1023/A:1010933404324.

[12] J. Kennedy, Particle swarm optimization, in: Encyclopedia of Machine Learning, Springer, 2010, pp. 760–766. doi:10.1109/ICNN.1995.488968.

[13] D. R. Jones, M. Schonlau, W. J. Welch, Efficient Global Optimization of Expensive Black-Box Functions, J. of Global Optimization 13 (4) (1998) 455–492. doi:10.1023/A:1008306431147.

[14] K. Crombecq, I. Couckuyt, D. Gorissen, T. Dhaene, "Space-Filling Sequential Design Strategies for Adaptive Surrogate Modelling", in B. H. V. Topping, Y. Tsompanakis, (Editors), "Proceedings of the First International Conference on Soft Computing Technology in Civil, Structural and Environmental Engineering", Civil-Comp Press, Stirlingshire, UK, Paper 50, 2009. doi:10.4203/ccp.92.50.

[15] K. Crombecq, E. Laermans, T. Dhaene, Efficient space-filling and non-collapsing sequential design strategies for simulation-based modeling, European Journal of Operational Research 214 (3) (2011) 683–696. doi:10.1016/j.ejor.2011.05.032.

[16] K. Crombecq, D. Gorissen, D. Deschrijver, T. Dhaene, A Novel Hybrid Sequential Design Strategy for Global Surrogate Modelling of Computer

Experiments, SIAM Journal of Scientific Computing 33 (4) (2010) 1948–1974. doi:10.1137/090761811.

[17] E. R. Van Dam, B. Husslage, D. Den Hertog, H. Melissen, Maximin Latin hypercube designs in two dimensions, Operations Research 55 (1) (2007) 158–169. doi:10.1287/opre.1060.0317.

[18] F. A. Viana, G. Venter, V. Balabanov, An algorithm for fast optimal Latin hypercube design of experiments, International journal for numerical methods in engineering 82 (2) (2010) 135–156. doi:10.1002/nme.2750.

[19] H. Akaike, A new look at the statistical model identification, Automatic Control, IEEE Transactions on 19 (6) (1974) 716–723. doi:10.1109/TAC.1974.1100705.

[20] D. Gorissen, I. Couckuyt, E. Laermans, T. Dhaene, Multiobjective global surrogate modeling, dealing with the 5-percent problem, Engineering with Computers 26 (1) (2010) 81–98. doi:10.1007/s00366-009-0138-1.

[21] D. Gorissen, I. Couckuyt, E. Laermans, T. Dhaene, Pareto-based multi-output metamodeling with active learning, in: Proceedings of the 11th International Conference on Engineering Applications of Neural Networks (EANN 2009), London, England, 2009.

[22] J. Sacks, W. J. Welch, T. J. Mitchell, H. P. Wynn, Design and analysis of computer experiments, Statistical science (1989) 409–423. doi:10.1214/ss/1177012420.

[23] D. C. Montgomery, Design and analysis of experiments, 5th Edition, John Wiley & Sons, 2001.

[24] D. Deschrijver, K. Crombecq, H. M. Nguyen, T. Dhaene, Adaptive Sampling Algorithm for Macromodeling of Parameterized S-Parameter Responses, IEEE Transactions on Microwave Theory and Techniques 59 (1) (2011) 39–45. doi:10.1109/TMTT.2010.2090407.

[25] T. Anderson, The Theory and Practice of Online Learning, 2nd Edition, Athabasca University Press, Canada, 2008.

[26] J. Aernouts, I. Couckuyt, K. Crombecq, J. J. Dirckx, Elastic characterization of membranes with a complex shape using point indentation

measurements and inverse modelling, International Journal of Engineering Science 48 (6) (2010) 599–611. doi:10.1016/j.ijengsci.2010.02.001.

[27] D. Stephens, D. Gorissen, K. Crombecq, T. Dhaene, Surrogate based sensitivity analysis of process equipment, Applied Mathematical Modelling 35 (4) (2011) 1676–1687. doi:10.1016/j.apm.2010.09.044.

[28] S. Koziel, L. Leifsson, Surrogate-Based Modeling and Optimization: Applications in Engineering, SpringerLink : Bücher, Springer New York, 2013.

[29] D. Deschrijver, F. Vanhee, D. Pissoort, T. Dhaene, Automated near-field scanning algorithm for the EMC analysis of electronic devices, IEEE Transactions on Electromagnetic Compatibility 54 (3) (2012) 502–510. doi:10.1109/TEMC.2011.2163821.

[30] S. Aerts, D. Deschrijver, W. Joseph, L. Verloock, F. Goeminne, L. Martens, T. Dhaene, Exposure assessment of mobile phone base station radiation in an outdoor environment using sequential surrogate modeling, Bioelectromagnetics 34 (4) (2013) 300–311. doi:10.1002/bem.21764.

[31] D. A. Cohn, Z. Ghahramani, M. I. Jordan, Active learning with statistical models, Journal of artificial intelligence research 4 (1996) 129–145. doi:10.1613/jair.295.

[32] R. M. Castro, Active learning and adaptive sampling for non-parametric inference, Ph.D. thesis, University of Wisconsin at Madison (2007).

[33] B. Settles, Active learning literature survey, University of Wisconsin, Madison 52 (55-66) (2010) 11.

[34] B. Settles, Active learning, Vol. 6, Morgan & Claypool Publishers, 2012. doi:10.2200/S00429ED1V01Y201207AIM018.

[35] N. Ailon, Active learning ranking from pairwise preferences with almost optimal query complexity, in: Advances in Neural Information Processing Systems, 2011, pp. 810–818.

[36] K. Trapeznikov, V. Saligrama, Supervised sequential classification under budget constraints, in: Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, 2013, pp. 581–589.

[37] A. Carpentier, M. Valko, Simple regret for infinitely many armed bandits., in: F. R. Bach, D. M. Blei, (Editors), Proceedings of the 31st International Conference on Machine Learning (ICML-15), Vol. 37, 2015, pp. 1133–1141.

[38] K. Jamieson, The Analysis of Adaptive Data Collection Methods for Machine Learning, Ph.D. thesis, UW-Madison (2014).

[39] S. J. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, 2nd Edition, Pearson Education, 2003.

[40] D. Gorissen, D. Deschrijver, T. Dhaene, D. D. Zutter, A Software Framework for Automated Behavioral Modeling of Electronic Devices [Application Notes], IEEE Microwave Magazine 13 (6) (2012) 102–118. doi:10.1109/MMM.2012.2205836.

[41] P. Singh, D. Deschrijver, D. Pissoort, T. Dhaene, Accurate hotspot localization by sampling the near-field pattern of electronic devices, IEEE Transactions on Electromagnetic Compatibility 55 (6) (2013) 1365–1368. doi:10.1109/TEMC.2013.2265158.

[42] Y. Lin, An Efficient Robust Concept Exploration Method and Sequential Exploratory Experimental Design, Ph.D. thesis, Georgia Institute of Technology (2004).

[43] Y. Zhao, C. Xu, Y. Cao, Research on query-by-committee method of active learning and application, in: Advanced Data Mining and Applications, Springer, 2006, pp. 985–991. doi:10.1007/11811305.

[44] R. Burbidge, J. J. Rowland, R. D. King, Active learning for regression based on query by committee, in: Intelligent Data Engineering and Automated Learning-IDEAL 2007, Springer, 2007, pp. 209–218.

[45] W. Hendrickx, T. Dhaene, Sequential design and rational metamodelling, in: M. Kuhl, N. M. Steiger, F. B. Armstrong, J. A. Joines, (Editors), Proceedings of the 2005 Winter Simulation Conference, 2005, pp. 290–298. doi:10.1109/WSC.2005.1574263.

[46] A. Basudhar, S. Missoum, A. H. Sanchez, Limit state function identification using support vector machines for discontinuous responses

and disjoint failure domains, Probabilistic Engineering Mechanics 23 (1) (2008) 1–11. doi:10.1016/j.probengmech.2007.08.004.

[47] A. I. Forrester, A. J. Keane, Recent advances in surrogate-based optimization, Progress in Aerospace Sciences 45 (1) (2009) 50–79. doi:10.1016/j.paerosci.2008.11.001.

[48] F. Hickernell, A generalized discrepancy and quadrature error bound, Mathematics of Computation of the American Mathematical Society 67 (221) (1998) 299–322. doi:10.1090/S0025-5718-98-00894-1.

[49] R. Jin, W. Chen, A. Sudjianto, An efficient algorithm for constructing optimal design of computer experiments, Journal of Statistical Planning and Inference 134 (1) (2005) 268–287. doi:10.1016/j.jspi.2004.02.014.

[50] H. Niederreiter, Random Number Generation and quasi-Monte Carlo Methods, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992.

[51] B. G. M. Husslage, et al., Maximin designs for computer experiments, Tech. rep., Tilburg University (2006).

[52] P. Z. G. Qian, Nested Latin hypercube designs, Biometrika 96 (4) (2009) 957–970. doi:10.1093/biomet/asp045.

[53] J. van der Herten, I. Couckuyt, D. Deschrijver, T. Dhaene, A Fuzzy Hybrid Sequential Design Strategy for Global Surrogate Modeling of High-Dimensional Computer Experiments, SIAM Journal on Scientific Computing 37 (2) (2015) A1020–A1039. doi:10.1137/140962437.

[54] P. Singh, D. Deschrijver, D. Pissoort, T. Dhaene, Adaptive classification algorithm for EMC-compliance testing of electronic devices, Electronics Letters 49 (24) (2013) 1526–1528. doi:10.1049/el.2013.2766.

[55] K. Fang, Experimental design by uniform distribution, Acta Mathematice Applicatae Sinica 3 (1980) 363–372.

[56] A. Farhang-Mehr, S. Azarm, Bayesian meta-modelling of engineering design simulations: a sequential approach with adaptation to irregularities in the response behaviour, International Journal for Numerical Methods in Engineering 62 (15) (2005) 2104–2126. doi:10.1002/nme.1261.

[57] G. Turk, M. Levoy, Zippered polygon meshes from range images, in: Proceedings of the 21st annual conference on Computer graphics and interactive techniques, ACM, 1994, pp. 311–318.

[58] D. R. Jones, C. D. Perttunen, B. E. Stuckman, Lipschitzian optimization without the Lipschitz constant, Journal of Optimization Theory and Applications 79 (1) (1993) 157–181. doi:10.1007/BF00941892.

[59] M. T. Hagan, M. B. Menhaj, Training feedforward networks with the Marquardt algorithm, IEEE Transactions on Neural Networks 5 (6) (1994) 989–993. doi:10.1109/72.329697.

[60] R. Houthooft, J. Ruyssinck, J. van der Herten, S. Stijven, I. Couckuyt, B. Gadeyne, F. Ongenae, K. Colpaert, J. Decruyenaere, T. Dhaene, et al., Predictive modelling of survival and length of stay in critically ill patients using sequential organ failure scores, Artificial intelligence in medicine 63 (3) (2015) 191–207. doi:10.1016/j.artmed.2014.12.009.

[61] I. Couckuyt, D. Deschrijver, T. Dhaene, Fast calculation of multiobjective probability of improvement and expected improvement criteria for pareto optimization, Journal of Global Optimization (2013) 575–594. doi:10.1007/s10898-013-0118-2.

[62] K. Elsayed, Optimization of the cyclone separator geometry for minimum pressure drop using Co-Kriging, Powder Technology 269 (2015) 409–424. doi:10.1016/j.powtec.2015.09.003.

[63] Y. Xiong, W. Chen, D. Apley, X. Ding, A non-stationary covariance-based Kriging method for metamodelling in engineering design, International Journal for Numerical Methods in Engineering 71 (6) (2007) 733–756. doi:10.1002/nme.1969.

[64] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, Evolutionary Computation, IEEE Transactions on 6 (2) (2002) 182–197. doi:10.1109/4235.996017.